Please replace the paragraph beginning on page 2, line 14 with the following amended paragraph:

The Unified Modeling Language (UML) defines a standard notation for representing flow charts for business workflow processes. However, the UML notation cannot be reduced to a programming language for employing complicated workflow processes. Current business workflow software systems provide scheduling software that requires binding within the scheduling to couple the schedule to real world applications and technologies. Conventional schedules ~~requires~~ require code to couple the components of the schedule to application program interface (API) objects and/or server objects for interfacing the schedule with systems within each business or department involved in the business process. These types of schedule software require sophisticated programmers ~~in implementing~~ to implement the software for a given business workflow model. Furthermore, these types of schedule software require modification of each schedule for different technologies.

Please replace the paragraph beginning on page 3, line 13 with the following amended paragraph:

BB 11-27-06

A user is provided with a GUI schedule interface, which allows the user to create a schedule on a first side of the GUI and to define bindings on the other side of the GUI. During creation of the schedule, the scheduler program prohibits the user from creating a schedule that will deadlock the schedule by checking correctness of the schedule flow. In creating the binding, the GUI schedule program provides prompts for specifying interfaces and methods of the components being bound. A data flow connection sheet is provided based on the schedule messages and the binding component interfaces and methods. The data flow of messages is then defined by simply connecting the message ports to binding component interfaces to facilitate proper data flow between entities. It is to be appreciated that the separate data flow sheet could be combined into the GUI scheduler interface or provided via a pop up screen after defining binding component interfaces and methods.

BB 11-22-06

Please replace the paragraph beginning on page 22, line 27 with the following amended paragraph:

Every message has two system fields ("Result Status", int) and ("Sender", string). The user shall not be able to enter a default value for the "Result Status" or Sender field. In the Field display section of each message editor, Name and Data Type are not editable. Data Types are typically displayed as XML types, regardless of message editor. Field names are unique per message. For the name, each message editor has a combobox containing all existing messages to enable the user to say that this message is exactly another message. Each message editor will have a rename button to the right of the combobox to allow the user to rename the currently selected message. This rename/combox will be the same whether selected from the data or business logic pages. Message names are displayed as a textbox, on both the business logic and data pages. Unique name validation will be done when the user presses the "OK" button. This editor will allow the user to specify two schemas (.XML files). The Message fields frame appears identical to the Method message editor. The names and data types are populated from the top-level XML Elements in the Internal Schema specified in the Message Schema frame. The user will be able to select from all transactions on the page. With COM and Method bindings, the fields of the messages are automatically set to either the [in] or the [out] arguments of method they are pointed to in the port. With MSMQ bindings, the user will have to go into the message ~~editor~~ editor, specify an XML schema, and select nodes in the tree to populate the fields. With CALL bindings, the user will have to go and specify the ports, messages, and transactions being passed into the called schedule.